Defence Research and Development Canada
Recherche et développement pour la défense Canada

Path Tracking for Unmanned Ground Vehicle Navigation

Implementation and Adaptation of the Pure Pursuit Algorithm

J. Giesbrecht, D. Mackay, J. Collier, S. Verret
DRDC Suffield

Technical Memorandum
DRDC Suffield TM 2005-224
December 2005

Canada

Path Tracking for Unmanned Ground Vehicle Navigation

Implementation and Adaptation of the Pure Pursuit Algorithm

J. Giesbrecht, D. Mackay, J. Collier, S. Verret
DRDC Suffield

Defence R&D Canada – Suffield

Technical Memorandum

DRDC Suffield TM 2005-224

December 2005

Author

J. Giesbrecht

Approved by

D. M. Hanna
Head/AISS

Approved for release by

Dr. P. A. D'Agostino
Head/Document Review Panel

Abstract

Following user defined paths and seeking goal locations is fundamental to Autonomous Unmanned Ground Vehicle (UGV) navigation. This paper summarizes the current state of the art in robotic path tracking for Ackerman steered vehicles and presents results of implementation and adaptation of the Pure Pursuit algorithm at Defence R&D Canada – Suffield.

Résumé

La poursuite de parcours configurés par l'utilisateur et la recherche de la location des buts sont fondamentales à la navigation autonome des Véhicules terrestres sans pilotes (UGV). Cet article résume l'état actuel de l'art de localiser des parcours avec la robotique en utilisant des véhicules munis d'un système de direction reposant sur le principe Ackerman et présente l'implémentation et l'adaptation de l'algorithme *Pure Pursuit* à R & D pour la défense Canada – Suffield.

This page intentionally left blank.

Executive summary

Path Tracking for Unmanned Ground Vehicle Navigation

J. Giesbrecht, D. Mackay, J. Collier, S. Verret; DRDC Suffield TM 2005-224; Defence R&D Canada – Suffield; December 2005.

Background: Path tracking algorithms may be used in a variety of Unmanned Ground Vehicle(UGV) applications. Typically the vehicle follows high level waypoints spaced tens or hundreds of meters apart which may be provided by a user for a patrol mission. To increase the vehicle's abilities, other behaviours such as obstacle avoidance, path planning or leader/follower augment control. The path tracking algorithm should be flexible enough to have suitable application in all these scenarios, working in concert with the other algorithms being used.

A number of approaches exist to accomplish this task. Some simple applications use proportional or PID control on the heading error between the vehicle and the goal. More complex systems use a tracking point along the prescribed path so that the vehicle will attempt to adhere to the path intended by the user. Of these systems, Pure Pursuit was chosen for its accuracy, simplicity, adaptability and robustness.

Principal Results: The Pure Pursuit algorithm was implemented in four different ways at Defence R&D Canada – Suffield:

1. As a path tracker to follow the straight line between high level waypoints on a patrol mission.

2. To provide goal directedness to an obstacle avoidance behaviour.

3. To allow a follower vehicle to pursue a lead vehicle via GPS breadcrumbs.

4. As a path tracker for a detailed on-line autonomous planner.

Through trials undertaken on a full size UGV vehicle, Pure Pursuit was found to be effective and flexible in all these roles.

Significance of Results: The Pure Pursuit algorithm was stretched well beyond its intended usage by these applications. The vehicle position and heading information provided by GPS was erratic, and vehicle actuators were slow and inaccurate. The output of the algorithm was combined with an obstacle avoidance algorithm to create more automous behaviour. However, the obstacle avoidance algorithm would at times take the vehicle well off its intended path to keep the vehicle safe. Despite these handicaps, the Pure Pursuit algorithm continued to function in a stable and effective manner. It can be concluded that Pure Pursuit is useful for a wide variety of robotic applications.

Future Work: The stability of the algorithm still needs to be investigated at higher speeds, and a predictive step should be added to account for steering lag. Additionally, it was found that following the straight line path between high level user waypoints is ineffective. A better solution would be to interpolate a spline between waypoints for the Pure Pursuit algorithm to follow.

Sommaire

Path Tracking for Unmanned Ground Vehicle Navigation

J. Giesbrecht, D. Mackay, J. Collier, S. Verret; DRDC Suffield TM 2005-224;
R & D pour la défense Canada – Suffield; décembre 2005.

Contexte : Les algorithmes de poursuite d'un parcours peuvent être utilisés par une variété d'applications de Véhicules terrestres sans pilotes (UGV). Le véhicule suit normalement des points de cheminement des hauts niveaux espacés de dix à cent mètres pouvant être fournis par un utilisateur de la mission de patrouille. Pour être en mesure d'augmenter les capacités du véhicule, d'autres comportements augmentent le contrôle lesquels consistent à éviter un obstacle, à planifier un parcours ou à jouer le rôle du chef et de l'exécutant. L'algorithme de poursuite d'un parcours devrait être assez flexible pour pouvoir être appliqué de manière pertinente à tous ces scénarios, en travaillant de concert avec les autres algorithmes qui sont utilisés.

Il existe un certain nombre de méthodes pour accomplir cette tâche. Quelques applications simples utilisent une commande proportionnelle, intégrale et dérivée sur l'erreur de cap entre le véhicule et son but. Des systèmes plus complexes utilisent un point tracé le long du parcours prescrit de manière à ce que le véhicule tente d'adhérer au parcours destiné à l'utilisateur. Parmi ces systèmes, Pure Pursuit a été choisi pour son exactitude, sa simplicité, son adaptabilité et sa robustesse.

Les résultats principaux : L'algorithme Pure Pursuit a été implémenté de quatre différentes façons à R &D pour la défense Canada – Suffield :

1. Comme dispositif de poursuite de parcours visant à poursuivre une ligne droite entre les points de cheminement des hauts niveaux durant une mission de patrouille.

2. Pour fournir, à un comportement d'évitement d'obstacles, la direction vers le but.

3. Pour permettre à un véhicule de poursuite de suivre un véhicule de tête au moyen des pistes de navigation du GPS.

4. Comme dispositif de poursuite de parcours pour un planificateur autonome en ligne et détaillé.

Les essais effectués sur un véhicule UGV grandeur nature ont indiqué que Pure Pursuit était efficace et faisait preuve de flexibilité dans tous ces rôles.

La portée des résultats : L'algorithme Pure Pursuit a été étendu bien au-delà de l'utilisation qu'on lui réservait par ces applications. La position du véhicule et les renseignements sur le cap fournis par le GPS étaient erratiques et les actionneurs du véhicule étaient lents est inexacts. On a combiné le résultat de l'algorithme avec l'algorithme d'évitement des obstacles

pour créer un comportement plus autonome. Cependant, l'algorithme d'évitement des obstacles écartait quelquefois beaucoup le véhicule du parcours prévu pour assurer la sécurité du véhicule. Malgré ces handicaps, l'algorithme de Pure Pursuit continuait de fonctionner d'une manière stable et efficace. On peut en conclure que Pure Pursuit est utile à une grande variété d'applications robotiques.

Les travaux futurs : La stabilité de l'algorithme a encore besoin d'être examinée à des vitesses supérieures et il faudrait ajouter une étape de prédiction à cause du traînage dans le guidage. De plus, on a trouvé qu'il n'est pas efficace de suivre un parcours en ligne droite entre les points de cheminement des hauts niveaux fournis par l'utilisateur. Une meilleure solution serait d'interpoler un spline entre les points de cheminement que suivrait l'algorithme Pure Pursuit.

Table of contents

Abstract i

Résumé i

Executive summary iii

Sommaire v

Table of contents vii

List of figures viii

1 Introduction 1

2 Background 1

3 Pure Pursuit Algorithm 3

4 Implementation and Results 6

4.1 Waypoint Navigation 6

4.2 Navigation Behaviour 8

4.3 Leader/Follower 9

4.4 Path Planning 10

4.5 Vehicle Control 11

5 Results 13

5.1 Lookahead Distance 13

5.2 Adaptive Lookahead 14

5.3 Radial Tolerance of Waypoints 15

5.4 Leader/Follower Behaviour 15

5.5 Waypoint Following 16

6 Conclusions 16

7 Future Work 18

References 19

List of figures

Figure 1: Various ways of defining the path. 2

Figure 2: Direct and indirect PID control for goal seeking. 3

Figure 3: Geometry of the Pure Pursuit algorithm showing the path to be tracked(left) and the calculated steering curvature(right). The curvature of the arc indicates a circle of radius r. Parameter l is the lookahead distance, with x and y defining the position of the lookahead point relative to the vehicle. 4

Figure 4: Sequential vehicle positions tracking a straight line path between two waypoints. 5

Figure 5: A high level path supplied via a user control station. 7

Figure 6: Adaptive lookahead can be used to make the algorithm more stable. . . . 8

Figure 7: The arbitration structure used to combine Pure Pursuit and Obstacle Avoidance votes. 9

Figure 8: A selection of candidate arcs, colored to indicate those more desirable to follow the intended path. 10

Figure 9: To follow a lead vehicle, the path to follow is defined as the straight line between the last known leader position, and the simultaneous follower position. 10

Figure 10: The Raptor UGV used to test the implementation of the Pure Pursuit algorithm. 12

Figure 11: The Ackerman geometry used to control the Raptor vehicle. 13

Figure 12: The effect of lookahead distance on performance when returning to a straight path (L = 1, 3 and 6 meters). 14

Figure 13: The effect of using adaptive lookahead to create more stable control (L = 1m). 14

Figure 14: The effect changing the Waypoint Tolerance 15

Figure 15: Paths of two vehicles, one following the other. Note the man driven vehicle trajectory appears smoother than the autonomous vehicle — evidence of straightline trajectories used between waypoints. 16

Figure 16: Path taken by the autonomous Raptor on a 2.5km waypoint patrol. The start section down the straight dirt path is at the top. 17

This page intentionally left blank.

1 Introduction

Under the Autonomous Land Systems project, scientists at Defence R&D Canada – Suffield have been investigating the guidance and control of Unmanned Ground Vehicles (UGVs). Autonomous vehicle navigation, the facility of moving from one position to another without operator supervision, is a key component of a UGV system. Goal-seeking and the ability to follow a pre-defined path supplied by a human operator are subsumed by this facility. This report summarizes the implementation of the Pure Pursuit algorithm [4] used to accomplish goal-seeking and path tracking.

Path tracking can profitably be employed in a variety of UGV applications. In a patrol mission scenario, the UGV may be tasked by a human operator to follow a series of high level waypoints, typically spaced tens to hundreds of meters apart. To follow such a coarse path reliably requires the use of on board sensing systems, analyzing the terrain around the UGV, and systems encompassing obstacle avoidance, path planning and/or tactical behaviours to negotiate the terrain between successive waypoints. In another scenario, the robot may be tasked to follow a lead vehicle or a more detailed path supplied by a human operator. A path tracking algorithm should be flexible enough to function in all of these scenarios, working in concert with the other algorithms being used.

While applicable to the above scenarios, the path tracking algorithm should also be robust to changing operating conditions. The errors in the estimates of vehicle heading, position, and speed input to the algorithm may all vary widely over time, potentially discontinuously. An operator may assign only a single high level goal, or a very detailed track for the vehicle to follow. The same algorithm may be used to control vehicles comprising a variety of sizes and configurations. A good path tracking algorithm should perform adequately under all of these conditions.

The remainder of this document contains some background into the general area of path tracking with a literature review of current methods. Next, the Defence R&D Canada – Suffield implementation of the Pure Pursuit algorithm is described. Then, the performance of the algorithm in a variety of scenarios is presented, and finally, future work is proposed to address some of the deficiencies observed in the performance of the algorithm.

2 Background

Path tracking uses positional information, typically obtained from GPS, IMU, and vehicle odometry, to control vehicle speed and steering to follow a specified path. In principle, a path could be defined as a continuous function. In practice, however, it is discretized, and described either as a series of straight line segments between interim goal waypoints or as a series of closely spaced nodes, as shown in Figure 1. If it is specified as a series of straight line segments, the algorithm will attempt to follow these straight lines. If it is specified as a series of closely spaced nodes, the algorithm will only attempt to attain the next node, whereas for a series of straight line segments, it will attempt to adhere to the

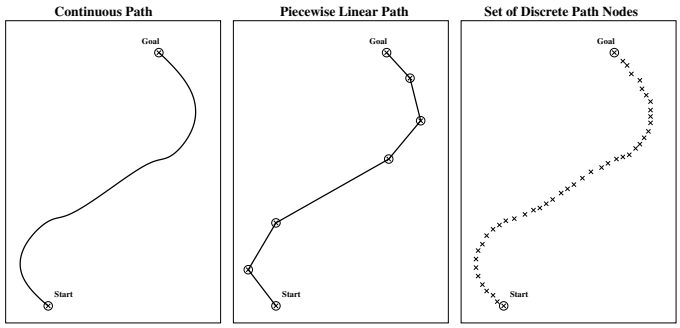

Figure 1: *Various ways of defining the path.*

lines described. This has no impact on either the formulation nor the implementation of a path tracking algorithm, since the target device is a digital computer, the path will be discretized at some point regardless of the input path description.

There are many ways to track paths. If a path is given as a set of waypoints then the simplest is proportional control on the error between the current heading and the heading to the next waypoint, as shown in Figure 2. This method provides goal directedness for many basic obstacle avoidance algorithms [1]. Because this method provides discontinuous control when switching from the current waypoint to the next waypoint as the interim goal, a Proportional-Integral-Derivative (PID) control loop is often used to correct the error to the goal heading.

Even in the absence of obstacle avoidance manoeuvres, the basic heading-to-goal controller as described above is incapable of accurately tracking a path specified by a series of waypoints. In the event that the vehicle is directed away from the path to avoid an obstacle in its way, having passed the obstacle it will turn back towards the path heading straight towards the interim goal directly from its current location. No attempt is made to track the path itself. A simple expedient which improves path tracking is to steer towards a point on the path a fixed lookahead distance from the vehicle rather than steering towards the next waypoint or the goal position itself. The lookahead point, shown in Figure 2, slides along the path a fixed distance ahead of the vehicle. A PID loop can then be used to control the error between the current heading and the heading to the tracking "carrot", adhering more closely to the intended path.

In general, PID control on the heading error is not ideal because it doesn't take into account the steering geometry of the vehicle. Additionally, it can be time consuming to tune the proportional, derivative and integral terms to generate desired behaviour, and has an inherent lateral path error around curves.

Using the concept of a lookahead point, a number of other controllers have been implemented. The Control Theory approach [5] executes proportional control on the heading as well as parallel displacement from the lookahead point. It is simple, but was shown to be an inferior method. The Quintic Polynomial approach [5] fits a polynomial that originates at the current vehicle position and heading, and ends at the lookahead point heading and

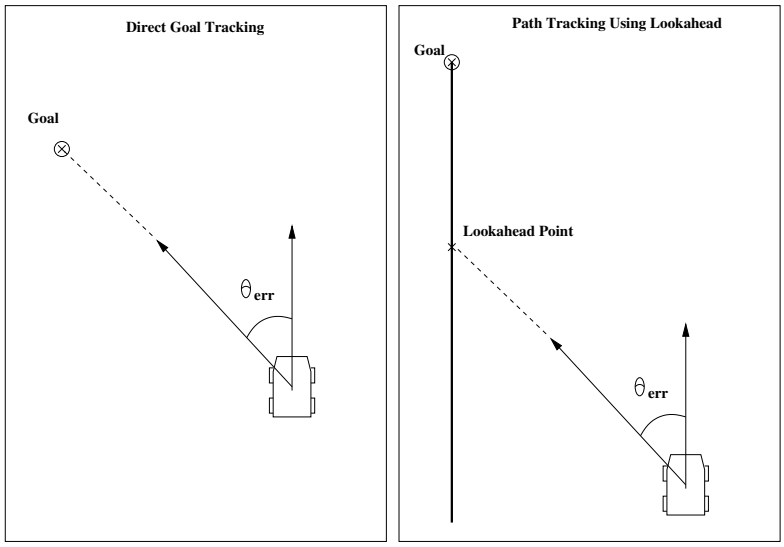

Figure 2: *Direct and indirect PID control for goal seeking.*

position. Unfortunately, the system was too complex to be practical, and performance was found to be overly dependant on lookahead distance. The Quadratic Curve method [6] is similar to the quintic polynomial except that it makes use of a quadratic curve instead. The Zhang tracker [2, 3], intended for differentially steered robots, has proved effective, but causes oscillations on curved paths. Finally, the Vector Pursuit method [7], which has its basis in screw theory, expands the Pure Pursuit method to operate not only on the displacement to the lookahead point, but also on the desired heading. It seems to perform as well or better than Pure Pursuit, but is more complex.

Among all these methods, the Pure Pursuit algorithm was chosen for its effectiveness and simplicity. Given that the algorithm would be required to operate under a number of different scenarios, it was desirable that it be robust and not so complex as to need a great deal of implementation effort. From the experiences at DRDC, the algorithm has proved to be very useful.

3 Pure Pursuit Algorithm

The Pure Pursuit algorithm [8, 5, 4, 9, 10] was devised to compute the arc necessary to return a vehicle back onto a path. It computes the curvature of an arc that a vehicle must follow to bring it from its current position to some goal position, where the goal is chosen as some point along the path to be tracked. The algorithm iterates continuously, with the goal point sliding along the path, forming a smooth tracking trajectory, as shown in Figure 4. It operates in a fashion similar to the way humans drive, in which we fixate on a point some distance ahead on the road and attempt to follow it.

In Pure Pursuit, we consider a constant curvature arc connecting the current vehicle position

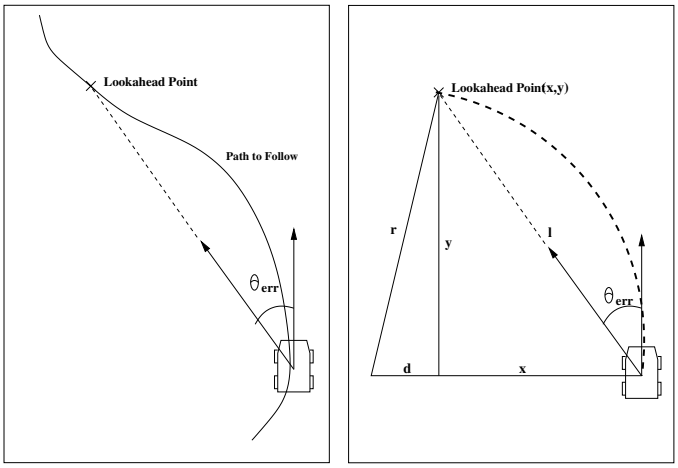

Figure 3: *Geometry of the Pure Pursuit algorithm showing the path to be tracked(left) and the calculated steering curvature(right). The curvature of the arc indicates a circle of radius r. Parameter l is the lookahead distance, with x and y defining the position of the lookahead point relative to the vehicle.*

and a point on the path a fixed distance ahead, called the lookahead distance (l), as shown in Figure 3. The lookahead point is one lookahead distance away from the vehicle, on the path to be followed.

From Pythagoras, we have

$$x^2 + y^2 = l^2 \tag{1}$$
$$d^2 + y^2 = r^2 \tag{2}$$

and from the figure

$$d = r - x. \tag{3}$$

Substituting Equation 3 into Equation 2 yields

$$(r - x)^2 + y^2 = r^2$$
$$x^2 + y^2 = 2rx. \tag{4}$$

And substituting Equation 4 into Equation 1 yields

$$2rx = l^2$$
$$r = \frac{l^2}{2x}. \tag{5}$$

The curvature of an arc is given as $\gamma = \frac{1}{r}$ so we can rewrite Equation 5 as

$$\gamma = \frac{2x}{l^2}. \tag{6}$$

Essentially, the Pure Pursuit algorithm is a proportional controller which operates on the error between the current vehicle heading and the heading to the goal point on the path.

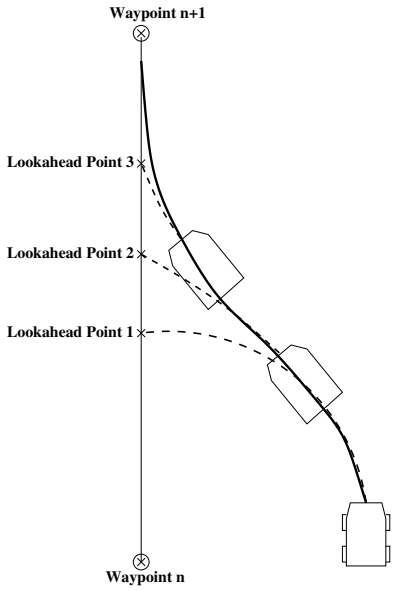

Figure 4: *Sequential vehicle positions tracking a straight line path between two waypoints.*

This can be seen by showing the formula for curvature in a different way. From Figure 3, we see that $sin(\theta_{err}) = \frac{x}{l}$, so for small heading errors $\theta_{err} \simeq \frac{x}{l}$. Substituting this into Equation 6, we get

$$\gamma = \frac{2\theta_{err}}{l}. \tag{7}$$

From this, we can see that the algorithm really only has a single parameter, the lookahead distance, l. This makes the algorithm exceedingly easy to implement and tune. Tuning this lookahead distance adjusts a number of performance characteristics. Having smaller lookahead distance forces the system to track the path more accurately, and increases the maximum curvature of a path that can be tracked. Additionally, a smaller lookahead distance causes the vehicle to return to the path more aggressively when it is separated. However, there are a number of good reasons to make it longer:

- A longer lookahead distance reduces oscillations while tracking a path.
- For paths which have sharp turns, it allows the vehicle to begin turning before it reaches the curve, resulting in smoother trajectories.
- The commanded steering changes are less abrupt, important for vehicles with high steering lag or operating at higher speeds, as the resultant turns are less sharp.
- With a larger lookahead distance here is less overshoot when returning to the path from a large separation.

4 Implementation and Results

The Defence R&D Canada – Suffield implementation of the basic Pure Pursuit algorithm described in the previous section has been used in the following scenarios:

1. Waypoint Navigation - Travel between high level waypoints assigned by an operator via a graphical control station. These waypoints will typically be spaced at distances of tens or hundreds of meters.

2. Navigation Behaviour - Provide goal directedness in concert with an obstacle avoidance algorithm.

3. Leader/Follower - Allow a follower vehicle to track a leader vehicle via GPS breadcrumbs.

4. Path Planning - Allow a vehicle to follow a detailed path assigned in real-time by an on-line path planner.

Each of these scenarios required a unique implementation of the basic Pure Pursuit algorithm. The implementation details required for each of these scenarios will be described in the following sections.

4.1 Waypoint Navigation

A typical scenario using waypoint navigation has a mission commander tasking the robotic vehicle with a patrol mission; the path to be followed is described as a series of waypoints tens or hundreds of meters apart. The autonomous vehicle navigates between consecutive waypoints without operator intervention. An example path is shown in Figure 5, in which a set of waypoints (red circles) is shown on the map display of a DRDC control station. Waypoints are sent to the robot as a set of positions, in latitude and longitude coordinates, accompanied by a radial tolerance for each. For a waypoint to have been reached, the robot must pass it at a distance less than or equal to the radial tolerance for that waypoint. The straight line segment between the previous and next waypoint will be the path tracked by the algorithm, as shown in Figure 4.

The standard implementation of the Pure Pursuit algorithm in the literature uses a detailed path described as a set of closely spaced nodes. However, for many UGV applications, the user will not want to specify an extremely detailed path. In the waypoint implementation of Pure Pursuit described here, each of the waypoints passed to it is considered an interim goal location and the basic algorithm tracks the straight line segments between consecutive pairs of waypoints. A list of all the waypoints is maintained, however, at any given time the algorithm is only operating between the current and the next waypoints.

The source of positional information is not necessarily important to the algorithm, however, it does require information about both position and heading. The sensor used for this implementation was the Sokkia GSR 2600 GPS receiver, which outputs heading and

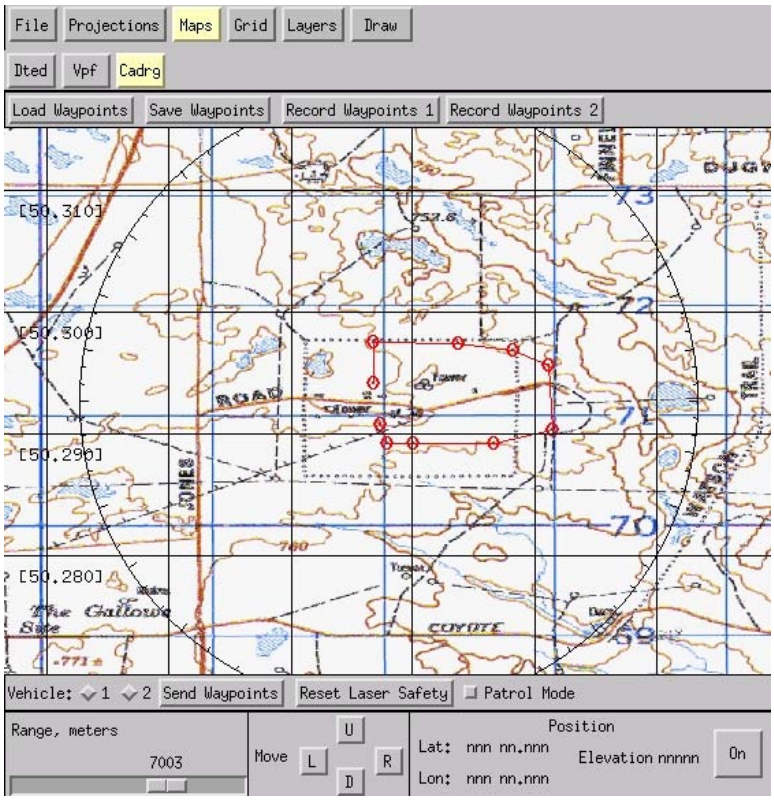

Figure 5: *A high level path supplied via a user control station.*

positional information from either differential or standard GPS readings at a rate of four times per second.

This implementation of the algorithm can be summarized as follows:

1. The implementation waits until it receives a path in the form of a waypoint list from the operator before moving.

2. Upon the receipt of a set of waypoints, each of the waypoints are converted from lat/long to northing/easting(meters).

3. Waypoint zero is set to the current location so the vehicle will move in a straight line from the start location to the first waypoint.

4. Each update from the GPS unit triggers the algorithm to check if it has reached the current waypoint, based upon the radial tolerance for that waypoint. If so, the next waypoint becomes the current waypoint, and the current waypoint becomes the last waypoint.

5. The algorithm finds the straight line between the last waypoint and the next waypoint.

6. It checks to see if it has gone past the next waypoint. If so, the geometry is reversed so that the algorithm will swing around to hit that waypoint.

7. It calculates the error between its current heading and the heading to the point one lookahead distance, l, ahead along the straight line path.

8. Using this error, it calculates the required curvature to reach that point as $\gamma = \frac{2\Theta_{err}}{l}$.

9. The algorithm repeats this procedure for each new positional update. If it reaches the final waypoint in the path, it checks the "patrol mode" or "path loop" flag status. If true, it will continue from the final waypoint to the first waypoint. If false, it halts the vehicle.

4.2 Navigation Behaviour

In the navigation behaviour implementation, an Obstacle Avoidance behaviour using range sensors and terrain mapping has been added to the basic Pure Pursuit in order to provide vehicle autonomy beyond that which blind waypoint following can offer. The path for the vehicle is still sent to the robot as discussed previously, but the output of the algorithm was modified to operate in conjunction with the Obstacle Avoidance behaviour.

The basic Pure Pursuit algorithm is less stable when the vehicle finds itself a long way off the path. Normally this would not be a problem when simply tracking a path, but the Obstacle Avoidance behaviour may move the vehicle away from the path a significant distance to avoid untraversable objects in the way. The addition of an *adaptive* lookahead distance, as described by Kelly [11] provides greater stability. The adaptive algorithm differs from the standard algorithm in only one respect, the lookahead distance is now no longer a fixed length, but varies with the distance between the vehicle and the path.

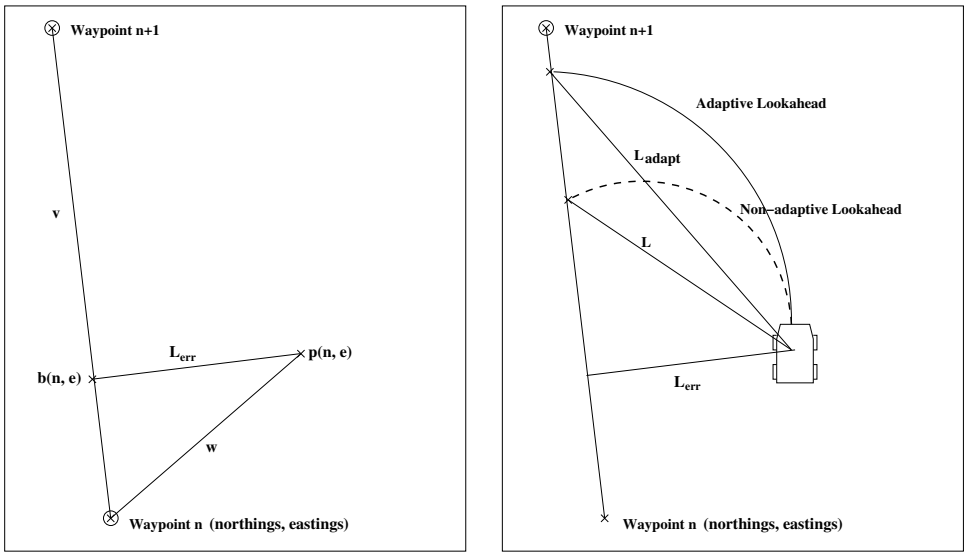

Figure 6: *Adaptive lookahead can be used to make the algorithm more stable.*

For a straight line path shown in Figure 6, this distance is found as follows. w is the vector from the last waypoint to the current position and v is the vector from the last waypoint to the next. Point p is the current position, n is the last waypoint, b is the point of projection of w on v. L_{err} is the distance between the vehicle position and the path, L is the fixed lookahead distance, and L_{adapt} is the adaptive lookahead distance used by the algorithm. Subscripts n and e indicate coordinates of northings and eastings respectively.

$$\begin{aligned} ||b|| &= \frac{w \cdot v}{v \cdot v} \\ b_n &= n_n + |b| \cdot v_n \\ b_e &= n_e + |b| \cdot v_e \\ L_{err} &= \sqrt{(p_n - b_n)^2 + (p_e - b_e)^2} \\ L_{adapt} &= L + L_{err} \end{aligned}$$

In order to combine the Obstacle Avoidance behaviour with the Pure Pursuit algorithm, an Arc Arbiter operates on the votes from each behaviour for a set of candidate arcs, shown in Figures 7 and 8. It combines votes from navigation behaviours to obtain a single steering command which is sent to the vehicle controller.

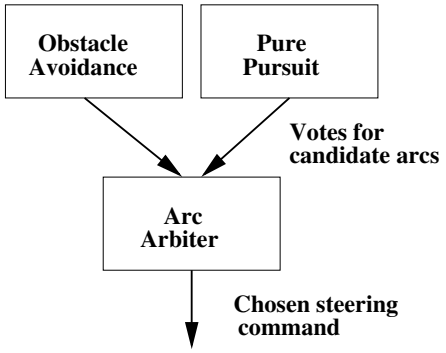

Figure 7: *The arbitration structure used to combine Pure Pursuit and Obstacle Avoidance votes.*

In order to accomplish this, a second major modification to the basic Pure Pursuit algorithm was made. To allow the Arc Arbiter to choose a steering command safe from obstacles yet still goal directed, the steering recommendations (votes) from the Pure Pursuit implementation are distributed on a Gaussian curve about the ideal steering angle. The Gaussian distribution and the set of candidate arcs are shown in Figure 8. A distribution factor σ can be adjusted by the user to change how widely the votes are distributed.

4.3 Leader/Follower

In the leader/follower implementation, Pure Pursuit is used to allow an autonomous vehicle to follow a second, human driven vehicle. Once more, the code used was the same Waypoint Navigation software described in Section 4.1.

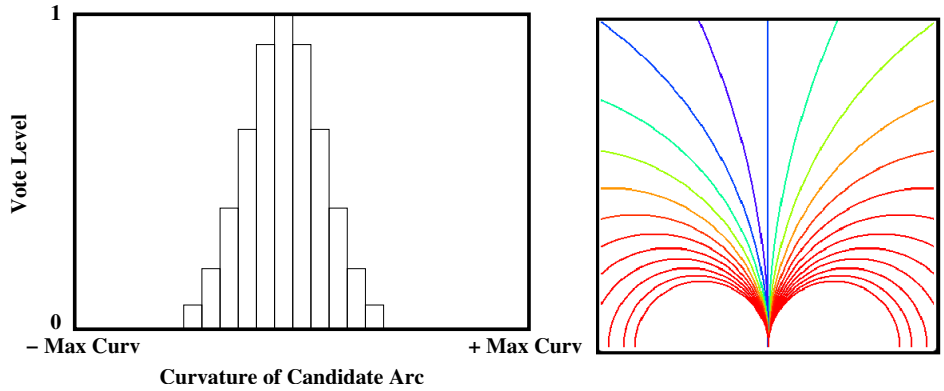

Figure 8: *A selection of candidate arcs, colored to indicate those more desirable to follow the intended path.*

However, instead of a complete path being passed to the Pure Pursuit Module, waypoints are passed only one at a time. Every few seconds, the leader passes, via radio network, a GPS breadcrumb to act as a new waypoint for the follower vehicle to track. Upon receipt, the leader's breadcrumb is set as the next waypoint for the follower vehicle to reach, and the follower's previous waypoint is set to its current position. In this way, the follower continually traces the straight line between its own position and the last known leader position, as shown in Figure 9.

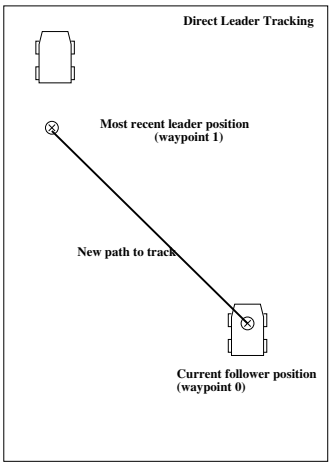

Figure 9: *To follow a lead vehicle, the path to follow is defined as the straight line between the last known leader position, and the simultaneous follower position.*

4.4 Path Planning

The final implementation of the Pure Pursuit algorithm was as a tracker to follow a complex path defined by an on-line autonomous path planner. This path planner, based upon the D* Lite algorithm [12], also uses the same high level waypoints obtained from the operator control station. The planner uses traversability map data accumulated from on-board

sensors to plan the most efficient, safe path to reach the next waypoint, rather than pursuing that waypoint directly. The planner also replans this path in real time as new sensor data is accumulated.

As discussed so far, the Pure Pursuit implementations followed the straight line between consecutive waypoints. However, in this implementation, Pure Pursuit must now track a complex path between consecutive waypoints comprised of a set of nodes which changes with every replanning episode. Further, since D* Lite also operates in concert with Obstacle Avoidance, the potential exists for vehicle to be driven well away from the planned path. The Defence R&D Canada – Suffield implementation of D* Lite uses the lateral offset from the path being tracked as a measure of success. If the lateral offset exceeds some threshold value, then the planner must replan from the vehicle's current location. Due to the relatively tight coupling between planning and tracking a path, the Defence R&D Canada – Suffield implementation of D* Lite uses its own internal version of Pure Pursuit. Like Obstacle Avoidance and the Pure Pursuit algorithm described earlier, D* Lite outputs an instance of the an arc vote set, distributed on a Gaussian curve as described in Section 4.2, to the Arc Arbiter. The situation is further complicated because the planner constructs a path in a planning reference frame, essentially differing from the world reference frame by an offset and a scale factor, and the Arc Arbiter expects an arc vote set in the vehicle's egocentric frame.

The algorithm can be summarized as:

1. Obtain the current vehicle pose estimate.
2. Find the path node, in the planning frame, closest to the current vehicle location.
3. Determine the goal node one lookahead distance from the current location. The goal node is found by moving up the path and calculating the distance between that path node and the vehicle's current location. It is possible that there are multiple points on the path one lookahead distance from the current vehicle location, and vehicle should steer towards the closest one in order to minimize the tracking error.
4. Transform the goal node location from the planning frame to the vehicle egocentric frame. The curvature γ is a function of the lateral offset x in the vehicle frame of the goal point and the lookahead distance to the goal point, as described earlier.
5. Calculate the curvature and set the arc votes appropriately.

4.5 Vehicle Control

The test platform for our implementation of the Pure Pursuit algorithm was a modified Koyker Raptor (Figure 10), adapted for drive by wire. It has a 25 hp gasoline engine powering a 4x4 hydrostatic drivetrain. Included in the modifications was an MPC555 processor with PID control of steering and velocity.

The output of the Pure Pursuit algorithm is a curvature for the vehicle to travel, γ. The Arc Arbiter converts this curvature to a rotational and translational velocity for the vehicle

Figure 10: *The Raptor UGV used to test the implementation of the Pure Pursuit algorithm.*

to execute based upon a user defined translational velocity setpoint of the vehicle, v_{set}. The required rotational velocity, ω, is calculated from the curvature, γ, and the translational velocity setpoint, v_{set},

$$\begin{aligned} \gamma &= \frac{\omega}{v_{\text{set}}} \\ \omega &= \gamma \times v_{\text{set}} \end{aligned}$$

These values for ω and v_{set} are then passed to the vehicle controller, which convert them to a steering angle and drive speed. The average angle of the front wheels, δ, in a vehicle of wheelbase, L, with Ackerman steering (Figure 11) as a function of path radius of curvature, R, is given by Gillespie [13]. Gillespie has shown that

$$\delta = \frac{L}{R}$$

and from above

$$R = \frac{1}{\gamma} = \frac{v_{\text{set}}}{\omega}$$

therefore

$$\delta = \frac{L \times \omega}{v_{\text{set}}}$$

At this point the steering angle δ and the drive speed v_{set} are passed to the vehicle PID loops for control.

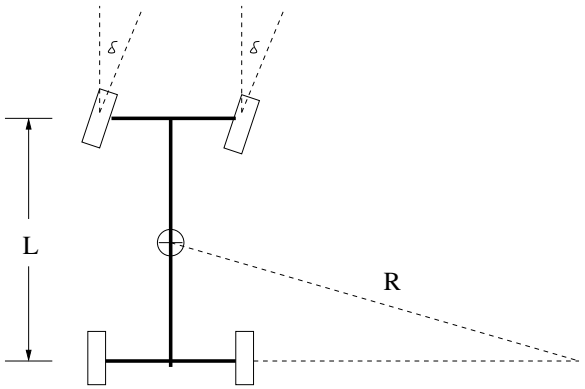

Figure 11: *The Ackerman geometry used to control the Raptor vehicle.*

5 Results

Using the Raptor platform discussed in Section 4.5, a number of tests were undertaken to verify the performance of the implementations of the Pure Pursuit algorithm. All of the tests were done with the algorithm following the straight line between high level waypoints, as described in Section 4.1. Some test results are presented below. When evaluating these tests, it is important to note that the only navigation sensor used was GPS position and heading. This means that positional data arrives discontinuously, and is prone to sudden jumps in value. In addition, it was observed that the heading information was inaccurate, prone to large fluctuations and lagged actual heading during turning. Despite these handicaps, the algorithm as implemented performed admirably.

5.1 Lookahead Distance

The first set of tests illustrate the effect of changing the lookahead distance on the algorithm. For this test, the last and next waypoints were given to the algorithm, defining a straight line for the vehicle to follow. The Raptor vehicle was then started at a distance offset from this line, to illustrate the step response of the algorithm.

Figure 12 shows the position of the vehicle in Lat/Lon coordinates. The vehicle begins at the right side of the graph and moves towards the left. The red line across the lower portion of the graph indicates the line it is attempting to follow. For these tests, the lookahead distance (non-adaptive) was set to 1m, 3m and 6m respectively. It can be seen that a shorter lookahead distance forces the algorithm to pursue the path more aggressively with less cumulative error, but results in sharper steering commands, and oscillation while following the path.

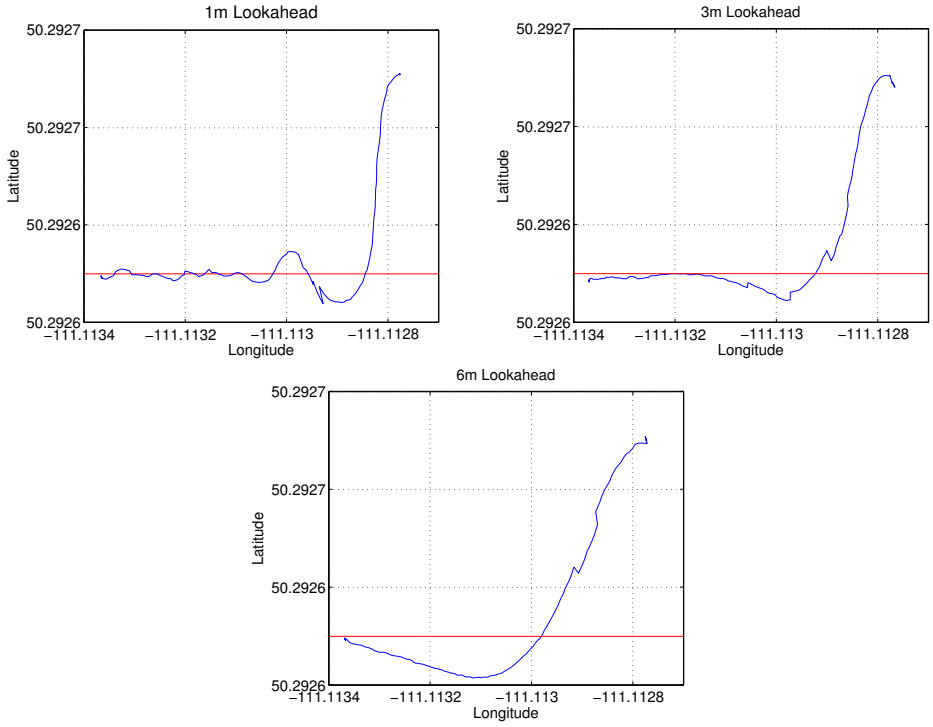

Figure 12: *The effect of lookahead distance on performance when returning to a straight path (L = 1, 3 and 6 meters).*

5.2 Adaptive Lookahead

The second test involved adding the adaptive lookahead parameter, as described in Section 4.2. The test setup is the same as in the first, except that for both graphs, a lookahead distance of 1 meter is used. It can be seen in Figure 13 that when the adaptive lookahead distance is used in the second graph, much better performance results. At the start the path is not pursued as aggressively with adaptive lookahead, but it is much smoother. By comparison with the non-adaptive lookahead results in Figure 12, the resultant trajectory can be made much smoother without sacrificing tracking accuracy.

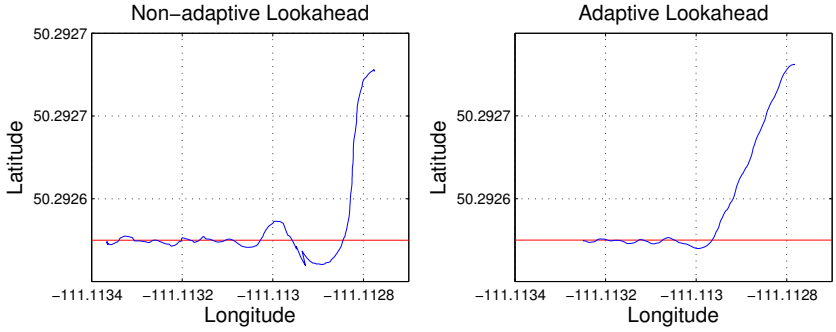

Figure 13: *The effect of using adaptive lookahead to create more stable control (L = 1m).*

5.3 Radial Tolerance of Waypoints

Another adjustable parameter unique to this implementation is the radial tolerance assigned to each waypoint, the distance away from the waypoint at which the robot considers that it has reached it. The problem of *subgoal obsession* is well known in robotics, where it is desirable for the robot to follow the path accurately, but not desirable for it to spend a large amount of time pursuing an individual subgoal. If the radial tolerance is set too large, the robot will move on to the next waypoint without really sticking to the path. However, if the radial tolerance is too small, then an Ackerman steered robot will overshoot the path where sharp corners occur at the waypoint. Additionally, with a small radial tolerance, if the robot has been drawn off the path by the Obstacle Avoidance behaviour, there is more chance of the robot swinging around to hit a waypoint, probably unnecessarily. In order to test the effect of radial tolerance, the Raptor platform was given a tight rectangular path to follow. This is somewhat difficult for the Raptor, as the PID loops in the steering controller cause a time lag in the execution of a steering angle causing path overshoot. Additionally, the platform has a rather large minimum turning radius of around 4 meters. The results for a 1m and 6m radial tolerance are shown in Figure 14 (lookahead distance was set to 3 meters). In the graph, when the radial tolerance is set too low (1 meter) for the vehicle, the path is overshot at the corners. With a more realistic radial tolerance, the robot does not approach the waypoint itself as closely, but is able to adhere to the path more accurately.

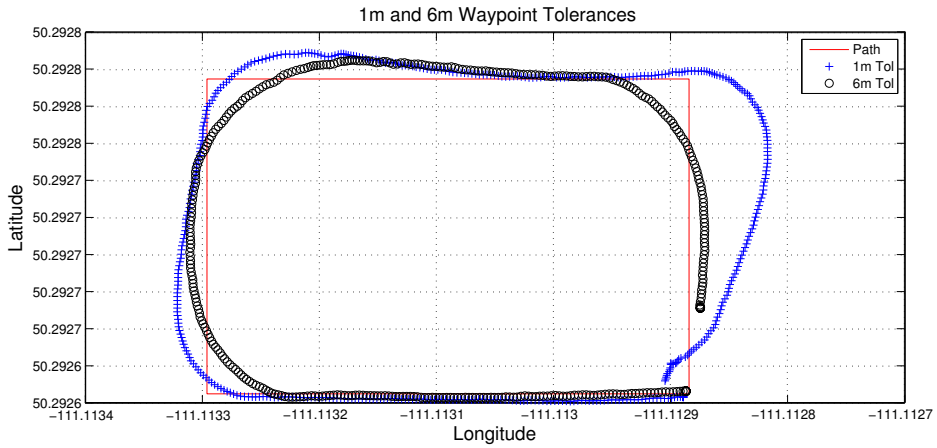

Figure 14: *The effect changing the Waypoint Tolerance*

5.4 Leader/Follower Behaviour

Tests of the Leader/Follower behaviour are shown in Figure 15. In this experiment, the human leader drove a number of amorphous paths in the test area, while GPS breadcrumbs were relayed to the autonomous follower vehicle every few seconds. This system is a simplistic approach, with the follower simply pursuing the straight line between itself and the last leader position. The system seemed to perform adequately, but it is evident that such a simplistic approach causes the vehicle to cut corners.

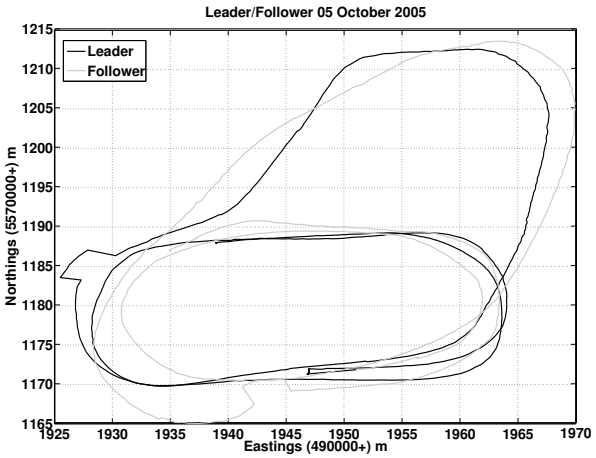

Figure 15: *Paths of two vehicles, one following the other. Note the man driven vehicle trajectory appears smoother than the autonomous vehicle — evidence of straightline trajectories used between waypoints.*

5.5 Waypoint Following

The final test presents the algorithm's reliability in following a high level, long distance path provided by a user control station for a patrol mission. The vehicle was tasked to drive down a dirt path for approximately 500 meters, and then patrol a large loop area (Figure 16). Total path length was approximately 2.5 km. The algorithm proved reliable over this long distance, and had no problem staying directly in the middle of the straight dirt path section. From experimentation for best performance, waypoints were spaced at 10 meters, and lookahead distance and waypoint tolerance were 3 meters and 5 meters respectively.

6 Conclusions

A number of lessons were learned in undertaking the various implementations of the Pure Pursuit algorithm. The most obvious is that the Pure Pursuit algorithm is extremely robust to poor sensing, poor actuation, combination with other control mechanisms, and is easily adapted for changing functionality.

Sensing used for these experiments consisted solely of a GPS unit providing positional and heading information. The tendency of GPS readings to hop as satellite geometry changes, caused the vehicle heading to change abruptly. When Differential GPS was unavailable the steering would tend to be erratic, especially with the vehicle close to the path, where a small shift in GPS postion or heading would cause a large shift in steering command. In addition, the software did not account for the offset of the GPS antenna from the centre of the turning radius of the vehicle. That being said, the algorithm remained stable at all times, still reaching the goal, despite the small steering oscillations caused by GPS inaccuracies. In fact, it even tracked paths well prior to a calibration of the steering, at

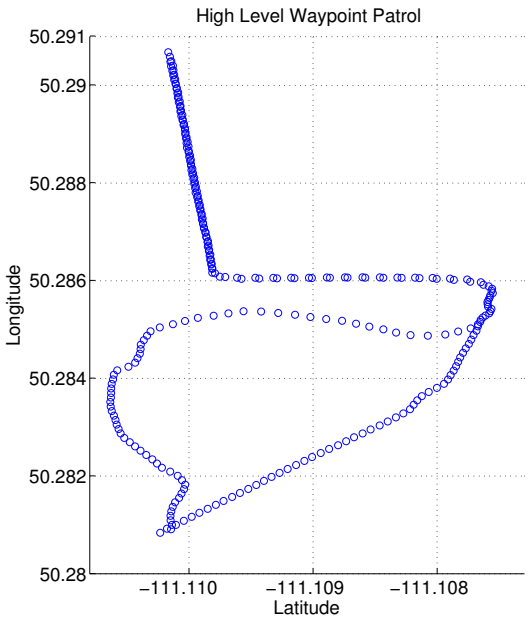

Figure 16: *Path taken by the autonomous Raptor on a 2.5km waypoint patrol. The start section down the straight dirt path is at the top.*

which time the PID loops were found to be out by 8 degrees.

Because each iteration of the Pure Pursuit algorithm is independent, it has a number of desirable qualities. It was found to work well when its output was smoothed to a Gaussian curve rather than the ideal steering angle, and also functions well under the influence of other behaviours such as obstacle avoidance. It also worked adequately with asynchronous updates of positional data and goal locations, having no requirement for regular position updates. It was also found that the adaptive lookahead is very desirable for applications in which the vehicle may get pulled a long distance off the path.

Finally, the limitations of requiring the robot to follow the straight line between consecutive waypoints became immediately apparent. This requirement necessitates the user to prescribe appropriate waypoint tolerances or risk the vehicle looping around to hit a waypoint it had already passed. It also meant that the algorithm could not anticipate a sharp corner at a waypoint and begin turning in advance of actually arriving at the waypoint.

Overall, however, Pure Pursuit was found to be a simple, accurate and robust algorithm for path tracking.

7 Future Work

From the lessons learned in implementing this algorithm, a number of further tasks should be undertaken to enrich this work.

Firstly, the straight line between waypoint method of path creation should be abandoned. A better approach would be to fit a spline between waypoints, or at least break the straight line segments down into a series of path nodes, so the algorithm could begin turning before hitting a waypoint. This would eliminate the need for a user to define waypoint tolerances and to worry about subgoal obsession.

Secondly, the implementations have not been analyzed over a wide variety of operating conditions. The tests so far were low speed (2-3 m/s). At higher speeds it may require additional code, such as a predictive step to account for steering lag, or an adaptive lookahead distance dependant on speed as well as distance from the path. It also would be beneficial to quantify performance as a function of sensor data, with regards to accuracy, smoothness and update rate of the data, and to test the algorithm with a different navigation sensor. It is assumed that performance would be enhanced.

Finally, in the Leader/Follower behaviour, the model used to create paths for the follower robot was too simplistic. A better approach would be to keep track of the path the lead robot took, rather than attempting to directly pursue its last known position.

References

[1] Simmons, R. (1996), The Curvature-Velocity Method for Local Obstacle Avoidance, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3375–3382.

[2] Zhang, Y., Velinsky, S., and Feng, X. (1997), On the Tracking Control of Differentially Steered Wheeled Mobile Robots, *Journal of Dynamic Systems, Measurement and Control*, 1, 455–461.

[3] Roth, S. and Batavia, P. (2002), Evaluating Path Tracker Performance for Outdoor Mobile Robot, In *Automation Technology for Off-Road Equipment*.

[4] Coulter, R. Craig (1992), Implementation of the Pure Pursuit Path Tracking Algorithm, (Tech Report CMU-RI-TR-92-01), Carnegie Mellon University.

[5] Amidi, O. (1990), Integrated Mobile Robot Control, Master's thesis, Carnegie Mellon University.

[6] Tipsuwan, Y. and Chow, M. (2004), Model Predictive Path Tracking Via Middleware for Networked Mobile over IP Network, In *Proceedings of the American Control Conference*.

[7] Wit, J., Crane, C., and Armstrong, D. (2004), Autonomous Ground Vehicle Path Tracking, *Journal of Robotic Systems*, 21(8), 439–449.

[8] Shin, D., Singh, S., and Shi, W. (1991), A Partitioned Control Scheme for Mobile Robot Path Tracking, *Proceedings of IEEE International Conference on Systems Engineering*, 1, 338–342.

[9] Murphy, K. (1992), Navigation and Retro-traverse on a Remotely Operated Vehicle, In *IEEE Sigapore International Conference on Intelligent Control and Instrumentation*.

[10] Murphy, K. (1994), Analysis of robotic vehicle steering and controller delay, In *Proceedings of 5th International Symposium on Robotics and Manufacturing*, pp. 631–636.

[11] Kelly, A. (1997), Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon, Ch. RANGER: Feedforward Control Approach to Autonomous Navigation, pp. 105–144, Kluwer Academic Publishers.

[12] Koenig, S. and Likhachev, M. (2002), D* Lite, *Proceedings of the National Conference on Artificial Intelligence*, pp. 476–483.

[13] Gillespie, T.D (1992), Fundamentals of Vehicle Dynamics, Society of Automotive Engineers.

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Suffield
PO Box 4000, Station Main, Medicine Hat, AB, Canada T1A 8K6

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

Path Tracking for Unmanned Ground Vehicle Navigation

4. AUTHORS (last name, first name, middle initial)

Giesbrecht, J.; Mackay, D.; Collier, J.; Verret, S.

5. DATE OF PUBLICATION (month and year of publication of document)

December 2005

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc).

32

6b. NO. OF REFS (total cited in document)

13

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

Defence R&D Canada – Suffield
PO Box 4000, Station Main, Medicine Hat, AB, Canada T1A 8K6

9a. PROJECT NO. (the applicable research and development project number under which the document was written. Specify whether project).

9b. GRANT OR CONTRACT NO. (if appropriate, the applicable number under which the document was written).

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique.)

DRDC Suffield TM 2005-224

10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Following user defined paths and seeking goal locations is fundamental to Autonomous Unmanned Ground Vehicle (UGV) navigation. This paper summarizes the current state of the art in robotic path tracking for Ackerman steered vehicles and presents results of implementation and adaptation of the Pure Pursuit algorithm at Defence R&D Canada – Suffield.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

path following
robotics
navigation